The following Listing of Claims will replace all prior versions, and listings, of claims in the present application:

Listing of Claims:

Claim 1 (Currently Amended): A method for the controlled execution of a program ~~(26)~~, the program ~~(26)~~ being intended for a virtual machine ~~(VM, VM')~~, on a portable data carrier ~~(10)~~, wherein

- the data carrier ~~(10)~~ has a processor ~~(12)~~ which executes at least a first and a second virtual machine ~~(VM, VM')~~,
- the program ~~(26)~~ is executed both by the first and by the second virtual machine ~~(VM, VM')~~,
- ~~the~~ an operating state of the first virtual machine ~~(VM)~~ and ~~the~~ an operating state of the second virtual machine ~~(VM')~~ are checked during execution of the program ~~(26)~~ for correspondence, and
- execution of the program ~~(26)~~ is aborted if a difference is found between the operating state of the first virtual machine ~~(VM)~~ and the operating state of the second virtual machine ~~(VM')~~.

Claim 2 (Currently Amended): A method according to claim 1, ~~characterised in that~~ wherein checking of the operating state of the first virtual machine ~~(VM)~~ and of the operating state of the second virtual machine ~~(VM')~~ for correspondence comprises checking whether the state of a program counter ~~(PC)~~ of the first virtual machine ~~(VM)~~ is the same as the state of a program counter ~~(PC')~~ of the second virtual machine ~~(VM')~~.

Claim 3 (Currently Amended): A method according to claim 1, wherein ~~or claim 2, characterised in that~~ checking of the operating state of the first virtual machine ~~(VM)~~ and of the operating state of the second virtual machine ~~(VM')~~ for correspondence comprises checking whether the level of a stack pointer ~~(SP)~~ of the first virtual machine ~~(VM)~~ is the same as the level of a stack pointer ~~(SP')~~ of the second virtual machine ~~(VM')~~.

Claim 4 (Currently Amended): A method according to claim 1, wherein ~~any one of claims 1 to 3, characterised in that~~ checking of the operating state of the first virtual machine ~~(VM)~~ and the operating state of the second virtual machine ~~(VM')~~ for correspondence comprises checking whether ~~the~~ a value of the most recent element ~~(@SP)~~ in a stack ~~(ST)~~ associated with

the first virtual machine ~~(VM)~~ is the same as ~~the~~ a value of the most recent element ~~(@SP')~~ in a stack ~~(ST')~~ associated with the second virtual machine ~~(VM')~~.

Claim 5 (Currently Amended): A method according to claim 1, wherein ~~any one of claims 1 to 4, characterised in that~~ checking of the operating state of the first virtual machine ~~(VM)~~ and of the operating state of the second virtual machine ~~(VM')~~ for correspondence is in each case performed after an instruction of the program ~~(26)~~ has been executed both by the first and by the second virtual machine ~~(VM, VM')~~.

Claim 6 (Currently Amended): A method according to claim 1, wherein ~~any one of claims 1 to 5, characterised in that~~ the first and the second virtual machine ~~(VM, VM')~~ access a common heap ~~(28)~~ in a non-volatile memory ~~(20)~~ of the data carrier ~~(10)~~.

Claim 7 (Currently Amended): A method according to claim 6, wherein ~~characterised in that~~, when an instruction of the program ~~(26)~~ that contains a write operation to the common heap ~~(28)~~ is being executed, ~~the~~ a write operation is performed only by the first virtual machine ~~(VM)~~.

Claim 8 (Currently Amended): A method according to claim 7, wherein ~~characterised in that~~ the instruction of the program ~~(26)~~ is executed first by the first virtual machine ~~(VM)~~ and then by the second virtual machine ~~(VM')~~, and, instead of performing the write operation, the second virtual machine ~~(VM')~~ checks whether ~~the~~ a value that is to be written is present in the heap ~~(28)~~ at the location that is to be written to.

Claim 9 (Currently Amended): A method according to claim 1, wherein ~~any one of claims 1 to 8, characterised in that~~ the program ~~(26)~~ is a *Java Card Applet* intended for execution by a JCVM (*Java Card Virtual Machine*).

Claim 10 (Currently Amended): A portable data carrier ~~(10), especially a chip card or a chip module~~, having a processor, ~~(12) and~~ an operating system ~~(22)~~, at least a first and a second virtual machine, and a program, wherein ~~the operating system (22) has program instructions for causing the processor (12) to perform a method according to any one of claims 1 to 9~~

- the processor executes both the first and second virtual machine,
- the program is executed both by the first and by the second virtual machine,
- the operating system controls the processor to check the operating state of the first virtual machine and the operating state of the second virtual machine during execution of the program for correspondence, and
- the operating system controls the processor to abort execution of the program if a difference is found between the operating state of the first virtual machine and the operating state of the second virtual machine.

Claim 11 (Canceled).

Claim 12 (New): A portable data carrier according to claim 10, wherein the data carrier is one of a chip card and a chip module.

Claim 13 (New): A portable data carrier according to claim 10, wherein checking of the operating state of the first virtual machine and of the operating state of the second virtual machine for correspondence comprises checking whether the state of a program counter of the first virtual machine is the same as the state of a program counter of the second virtual machine.

Claim 14 (New): A portable data carrier according to claim 10, wherein checking of the operating state of the first virtual machine and of the operating state of the second virtual machine for correspondence comprises checking whether the level of a stack pointer of the first virtual machine is the same as the level of a stack pointer of the second virtual machine.

Claim 15 (New): A portable data carrier according to claim 10, wherein checking of the operating state of the first virtual machine and the operating state of the second virtual machine for correspondence comprises checking whether the value of the most recent element in a stack associated with the first virtual machine is the same as the value of the most recent element in a stack associated with the second virtual machine.

Claim 16 (New): A portable data carrier according to claim 10, wherein checking of the operating state of the first virtual machine and of the operating state of the second virtual

machine for correspondence is in each case performed after an instruction of the program has been executed both by the first and by the second virtual machine.

Claim 17 (New): A portable data carrier according to claim 10, wherein the first and the second virtual machine access a common heap in a non-volatile memory of the data carrier.

Claim 18 (New): A portable data carrier according to claim 17, wherein, when an instruction of the program that contains a write operation to the common heap is being executed, the write operation is performed only by the first virtual machine.

Claim 19 (New): A portable data carrier according to claim 18, wherein the instruction of the program is executed first by the first virtual machine and then by the second virtual machine, and, instead of performing the write operation, the second virtual machine checks whether the value that is to be written is present in the heap at the location that is to be written to.

Claim 20 (New): A portable data carrier according to claim 10, wherein the program is a *Java Card Applet* intended for execution by a JCVM (*Java Card Virtual Machine*).

Claim 21 (New): A computer program product having program instructions for causing a processor of a portable data carrier to perform a method for the controlled execution of a program, the program being intended for a virtual machine, wherein

- the processor executes at least a first and a second virtual machine,
- the program is executed both by the first and by the second virtual machine,
- the operating state of the first virtual machine and the operating state of the second virtual machine are checked during execution of the program for correspondence, and
- execution of the program is aborted if a difference is found between the operating state of the first virtual machine and the operating state of the second virtual machine.

Claim 22 (New): A computer program product according to claim 21, wherein the first and the second virtual machine access a common heap in a non-volatile memory of the data carrier.

Claim 23 (New): A computer program product according to claim 22, wherein, when an instruction of the program that contains a write operation to the common heap is being executed, the write operation is performed only by the first virtual machine.

Claim 24 (New): A computer program product according to claim 23, wherein the instruction of the program is executed first by the first virtual machine and then by the second virtual machine, and, instead of performing the write operation, the second virtual machine checks whether the value that is to be written is present in the heap at the location that is to be written to.

Claim 25 (New): A computer program product according to claim 21, wherein the program is a *Java Card Applet* intended for execution by a JCVM (*Java Card Virtual Machine*).